Toy Problem

The toy problem shown in figure 1 below uses an implementation for gradient domain processing. The goal was to reconstruct the original image from its gradient values, plus one pixel intensity. Each pixel is solved for, minimizing the three equations below and solved using a least squares calculation. V is denoted as what is being solved and S is the source image.

minimize (v(x+1,y)-v(x,y) - (s(x+1,y)-s(x,y)))^2
minimize (v(x,y+1)-v(x,y) - (s(x,y+1)-s(x,y)))^2
minimize (v(0,0)-s(0,0))^2

The image below shows a side by side comparison of the two images, notating that it has no distinctions, and the root-mean-squared error is shown below as well.

Poisson Blending

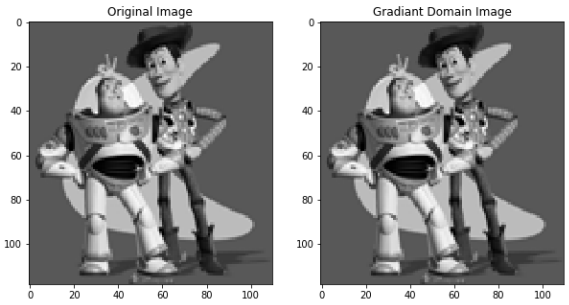

Figure 1: Toy Problem Root Mean Error is: 0.000317018500683

The Poisson Blending algorithm selects a boundary region in a source image and a location in a target image where the source image should be blended in. The source image is changed so that the indices of the source and target regions correspond, the rest of the target image remains unchanged. Linear algebra is used to solve for specific constraints for each pixel. The formula denotes v, the target pixel being minimized, and s as the source image pixel.

$$\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - (s_i - s_j))^2 + \sum_{i \in S, j \in N_i \cap \neg S} ((v_i - t_j) - (s_i - s_j))^2$$

In the implementation done, only the portion of the image that the source was overlaid over was solved for, known as the masked region. A small padded region was also added to the masked region so that pixels on the edges of the source region could easily be calculated without any coding errors. Another usage in the code was using a sparse matrix calculation. Because most of the numbers used in the matrix calculation are zero, a normal processor would not be able to handle the calculation. Finally, once each pixel is solved for in the masked region, they are input into the target image and the image is output. Figure 2 shows the dragon as the source image and the field as the target image. Figure 2 also shows the image both with and without the Poisson Blending.

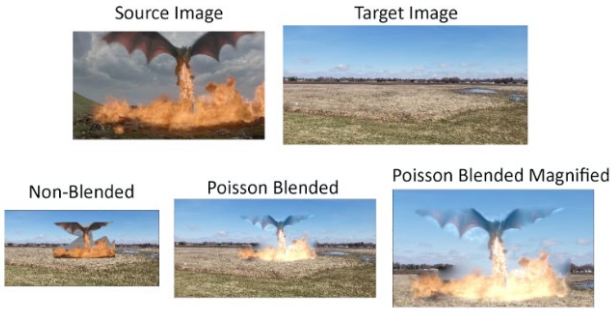

Figure 2: Poisson Blending

The final product shows a very well done blended image, especially the fire portion of it. The dragon itself looks more like the background blue sky but works in this case. In figure 3, we see another case of Poisson blending that does not work as well because the lion and the surrounding region in the grass have a different color. This causes parts of the lion to look black or very hard to see. In this case, the blending did not work well. Ideally the target image area and source image will have similar colors. Figure 4 shows the penguin being blended into the image. The penguin does well when blended into the snow region as shown, however, if the penguin was in the non-snow

region a similar effect would be seen as with the lion.

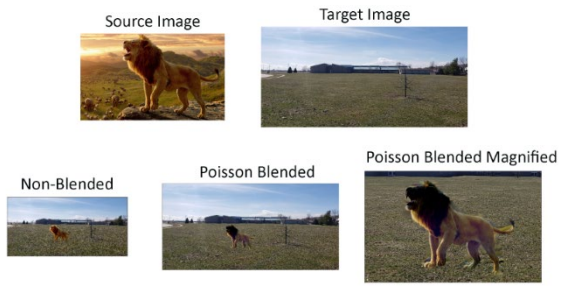

Figure 3: Poisson Blending

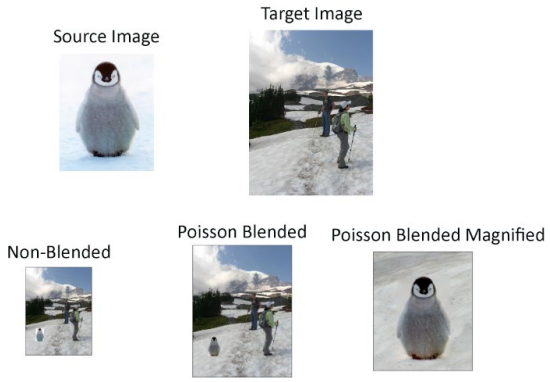

Figure 4: Poisson Blending

Mixed Gradients

The way mixed gradient is done is very similar to Poison Blending, with the exception that the gradient in the source or target region is used with the higher magnitude rather than just the source gradient. This is shown in the formula below. The value d_ij is denoted as that gradient with higher magnitude. Again, each pixel value v is minimized a system of linear equations is solved. The same concept in the code is shown, with the exception stated before of taking the higher gradient magnitude.

$$\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - d_{ij})^2 + \sum_{i \in S, j \in N_i \cap \neg S} ((v_i - t_j) - d_{ij})^2$$

Figure 5 shows the alligator in the pool using mixed gradients. Parts of the alligator's original gradients are preserved while still blending well into the pool. This gives the effect that the alligator is actually swimming in the pool.

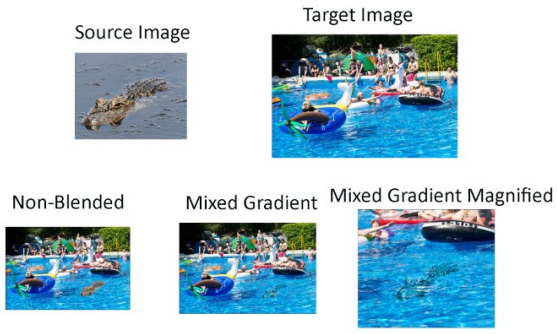

Figure 5: Mixed Gradient Blending

Bells & Whistles

Color2Gray

The same algorithm that was used for the toy problem was used similarly to convert two number images shown in figure 6. The goal was to reconstruct the image into grayscale using the highest value gradient RGB channel and to find the pixel value through gradient domain processing. Each pixel is solved by minimizing the three equations below and using a least squares calculation. V is denoted as what is being solved and S is the source image's highest gradient channel value.

minimize (v(x+1,y)-v(x,y) - (s(x+1,y)-s(x,y)))^2
minimize (v(x,y+1)-v(x,y) - (s(x,y+1)-s(x,y)))^2
minimize (v(0,0)-s(0,0))^2

Had the regular rgb2gray function been used, the number in the gray scaled image would be indistinguishable. This is due to the loss of important contrast information from the image.

Figure 6 shows a comparison of what it would look like if the rgb2gray function is used versus the function that was specifically created to capture the contrast information.

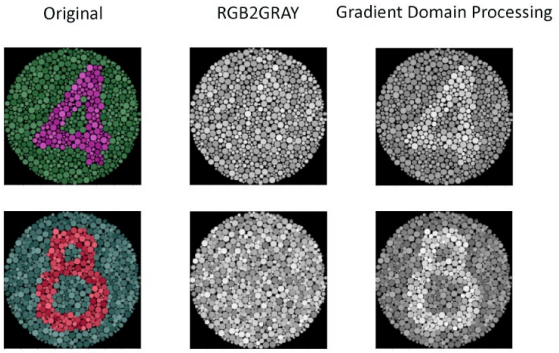

Figure 6: Color2Gray

Laplacian pyramid blending

Another form of blending is Laplacian pyramid blending that decomposes two images and uses a Laplacian pyramid to combine the two images. In the case of figure 7, two pictures of brothers were combined to create one face. As can be seen when blending the two together using Laplacian pyramid blending, a higher quality blend of the two images results compared to no blend at all. The code works as follows: each image creates its own respective— a Guassian and Laplacian pyramid. Then, each image half is blended together. For low frequencies, the image has a slow transition and for high frequencies the transition is sharp.

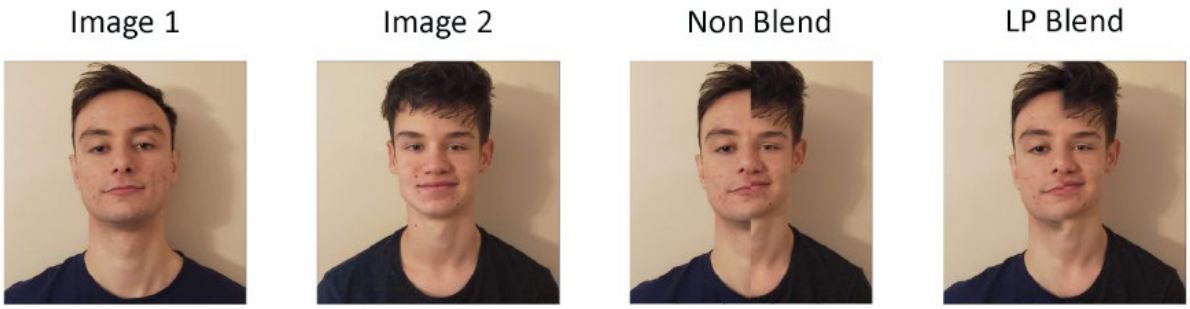

Figure 7: Laplacian Pyramid Blending

Sources

Alligator

https://s.hdnux.com/photos/37/51/10/8294941/5/rawImage.jpg

Dragon

https://i1.wp.com/xn--lacompaialibredebraavos-yhc.com/wp-content/uploads/2017/03/wlapawuuoeeqkfpk0udp.jpg?fit=1920%2C1080&ssl=1

Lion

https://www.nusantaratv.com/cdn/img/nusantaratv_750x_5deb447d6fb52.jpg

Pool

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSsf_EzNuUyqA8abEdnTwnkOrMMSgkODc8Va1ei0qZ9Ck-_usxY1A&s

Points

20/20 for implementation of toy image reconstruction

50/50 for Poisson blending implementation, results, and description: 30 pts for first result; 20 points for additional two (or more) results

20/20 for mixed gradients implementation and at least one result.

10/10 for quality of results and clarity of presentation.

20/20 Color2Gray

20/20 Laplacian pyramid blending

140/100 TOTAL